D4A Magazine

مجلة منتدى دلفي للعرب

Issue 02 - July 2010

العدد رقم : 02 لشهر يوليو 2010

iif vs if

ActionList

KOL-MCK
Components

إعادة الاستخدام في دلفي: الإطارات



IDR
Interactive Delphi Reconstructor

Interactive Delphi Reconstructor

تمرين هذا العدد
Flash Memory Filter/Locker

Overload

فهـــرس العدد



سجل النظام و الثغرات القانونية.



حين نقوم بقراءة اتفاقية الاستعمال خلال تنصيب البرامج و بالنقر على زر موافق نصبح مربوطين و ملزمين بالشروط التي تم ذكرها في الاتفاقية.

و من الطبيعي انه لا يسمح لنا بالتعديل في خيارات التسجيل للنسخ التجريبية بتاتا و أي تعديل يعتبر قرصنة للبرنامج و يعاقب عليه القانون.

و لكن ماذا عن التعديل في سجل النظام Windows Registry ؟

مثلا عندنا برنامج يقوم بحفظ عدد أيام استعماله في سجل النظام، و الاتفاقية الخاصة به تقول أن لنا الحق في استعماله بصفة Trial (مدة زمنية محدودة) فقط و علينا شراء النسخة الكاملة في حالة رغبة استعمال البرنامج بدون تحديدات.

التعديل المباشر في قيم البرنامج أو في قيم ملفاته يعتبر إخلال بالاتفاقية و لكن في نفس الوقت نملك كل الحق في التعديل في قيم سجل النظام لأنه عند شراء نسخة من نظام تشغيل Windows مثلا نملك الحق في حذف و تغيير مفاتيحه.
يعني حذف مفاتيح الـ Trial لأي برنامج قام بحفظها هو في سجل النظام لا يعتبر خرق للاتفاقية و القانون لا يعتبرنا مذنبين لان التعديل مس حق من حقوق حرية استغلال سجل النظام عند شراء نسخة من نظام التشغيل.

الكاتب: إدارة المنتدى

إعادة الاستخدام في دلفي ـ بقلم خالد شقروني

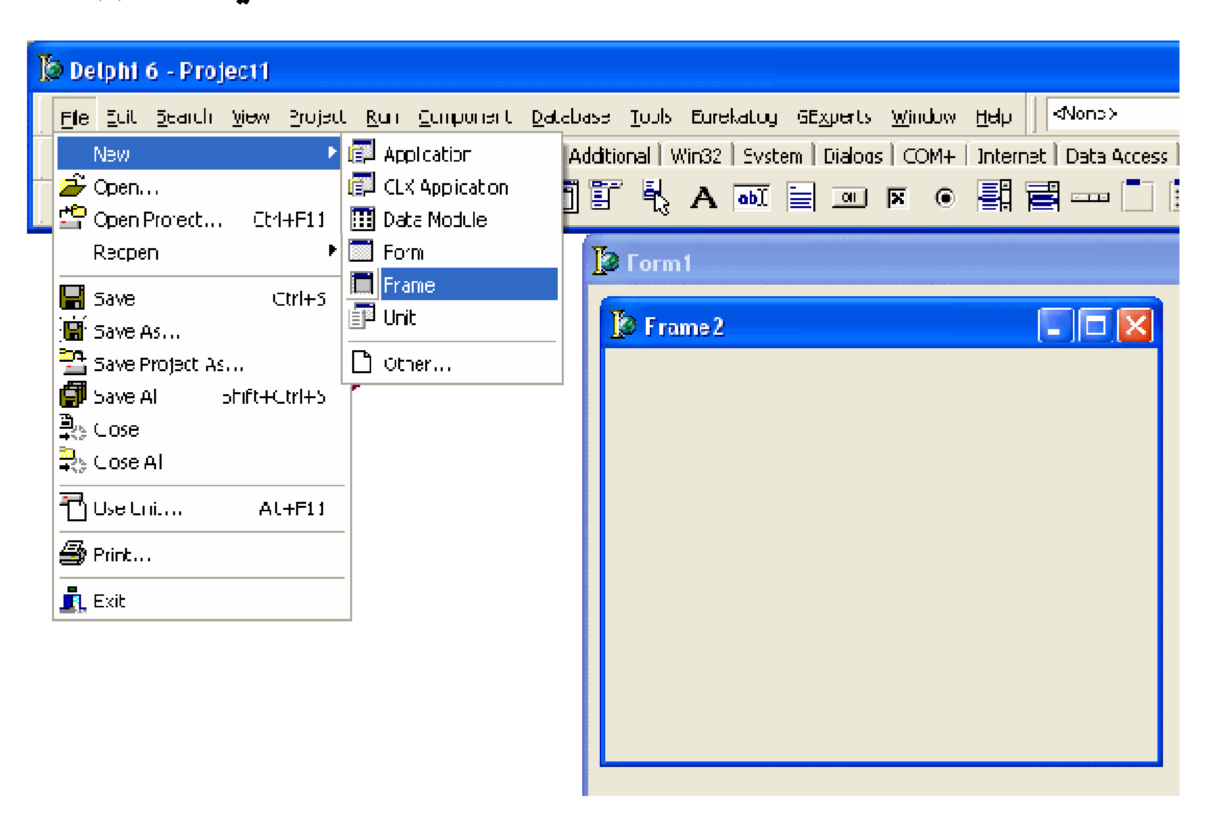
الجزء الثاني – الإطارات

الإطارات في دلفي عبارة عن حاوية لمجموعة مكونات يتم تحديد خصائصها و برمجة أحداثها، وتحفظ لكي يعاد استخدامها على نماذج شاشات التطبيق.

توفر هذه الإطارات إمكانية تصميم وبرمجة المكونات ضمن الإطار لمرة واحدة، ثم نعيد استخدامها لأكثر من مرة في شاشات البرنامج مثل أي مكون آخر.

لنبدأ باستطلاع كيفية إنشاء إطار Frame **واختبار سلوكياته، ثم ننظر في كيف يمكن لهذه الإطارات أن تفيدنا و كيف توفر علنا الكثير من الجهد و الوقت.**

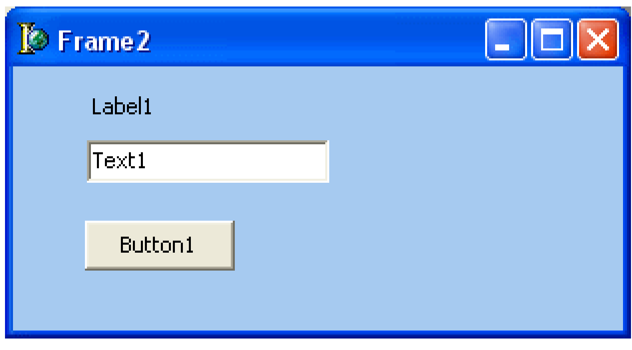
من خلال الأوامر File| New| Frame **نقوم بإنشاء نموذج إطار جديد كما هو في الشكل** (1)



الشكل (1)

وكما هو ظاهر فإن شاشة الإطار تشبه شاشة النموذج Form ، و سيكون الإطار جاهزا كي نضع عليه ما شئنا من مكونات.

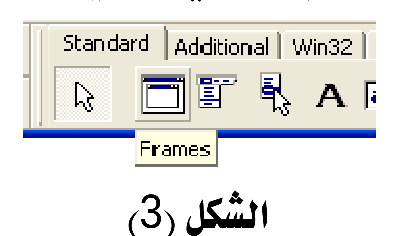
نضع على الإطار مكونات مثلا من نوع TEdit و TLable و TButton أيضا يمكن تغيير لون الإطار وحجمه. كما في الشكل (2)



الشكل (2)

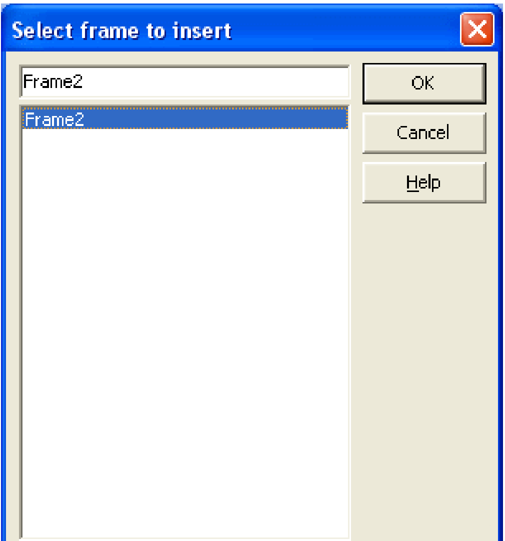
الآن بعد أن أنشأنا الإطار يمكننا مباشرة استخدامه، ووضعه على نموذج الشاشة.

في شريط المكونات وفي صفحة Standard نجد أول أيقونة فيه تمثل الإطارات كما في الشكل (3).



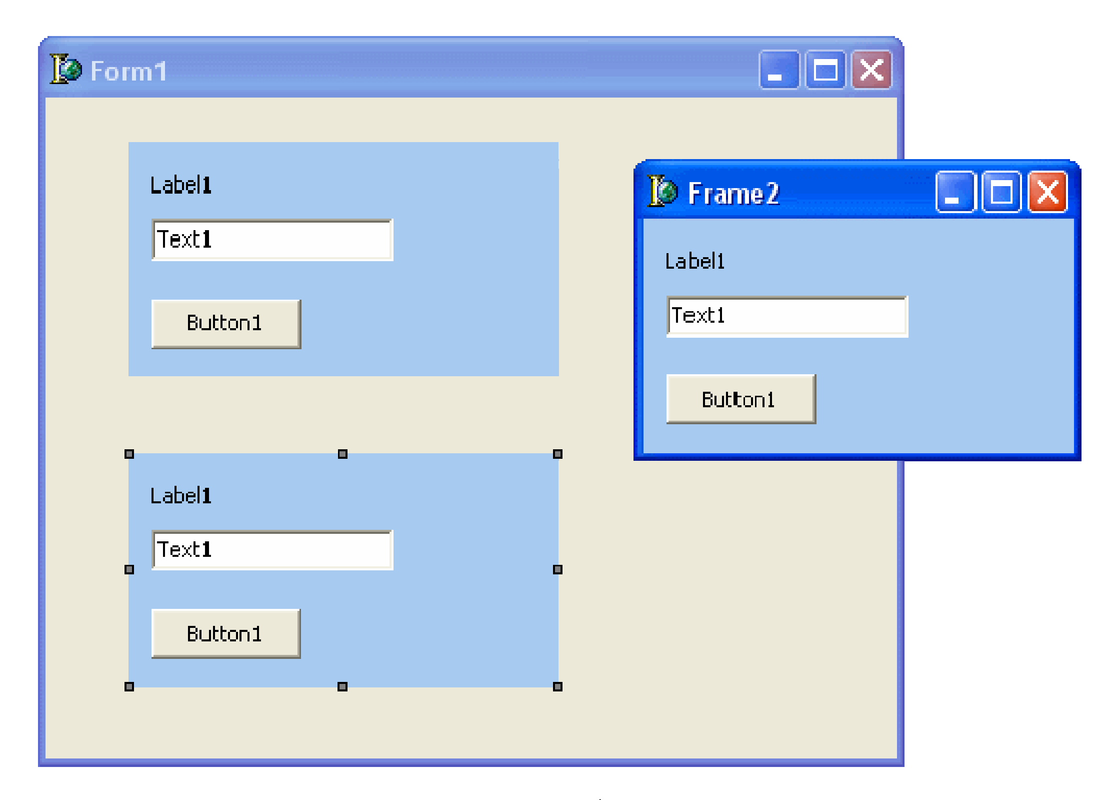
الشكل (3)

بواسطة الفأرة وبالضغط على هذه الأيقونة ثم على نموذج الشاشة Form سيظهر مربع به قائمة بالإطارات المتوفرة. الشكل (4)



الشكل (4)

نقوم باختيار الإطار من القائمة ثم موافق. فيتم حالا إدراج تجسيدا لهذا الإطار داخل نموذج الشاشة مثل أي مكون. يمكن أن نقوم بإدراج تجسيدا آخر للإطار كما في الشكل (5)



الشكل (5)

كما نرى في الشكل (5) لم نقم بإغلاق شاشة الإطار، وذلك لغرض مراقبة سلوكها.

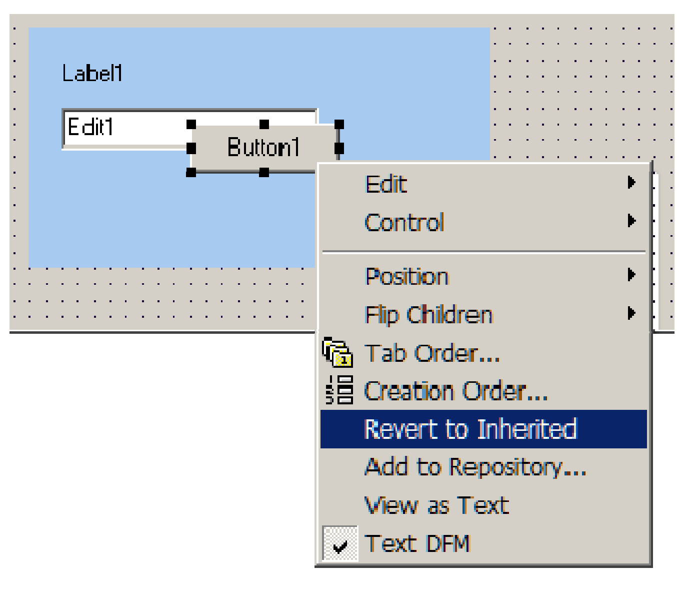
أي تجسيد لإطار يرث خصائص الإطار الأصلي

لو تأملنا شاشة الفورم سنجد أن تجسيد الإطار على الشاشة هو نسخة طبق الأصل من تصميم الإطار الأصلي. الآن، لو ذهبنا إلى الإطار الأصلي و غيرنا من خصائصه أو خصائص إحدى مكوناته (تغيير حجم الإطار، أو مكان الزر مثلا) سنلاحظ انعكاس هذا التغيير على الإطارات في الفورم فورا. أي أن تجسيدات الإطار في الفورم ترث خصائص الإطار الأصلي وتتأثر بأي تغيير يحدث على الإطار الأصلي.

نسخة الإطار يمكن تعديل خصائصها متجاوزة الإطار الأصلي

تجسيدات الإطار يمكن أن تتجاوز خصائص الإطار الأصلي؛ فلو ذهبنا إلى إحدى تجسدات الإطار في الفورم وحرّكنا مكان الزرأو الكتابة، فسنجدها تتغير وتستقل بخصائصها عن خصائص الإطار الأصلي.

لاسترجاع الخصائص الأصلية لأي مكون في تجسيد الإطار، نقوم باختيار المكون ثم بالزر الأيمن للفأرة نستدعي لائحة الأوامر الفرعية ونختار الأمر Revert to Inherited فيستعيد المكون خصائصه الموروثة التي كانت عليه في الإطار الأصلي كما هو مبين في الشكل (6)

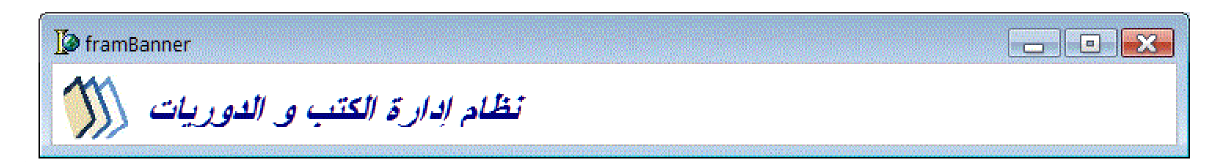


الشكل (6)

مثال عملي:

نريد لشاشات برنامجنا أن يكون لها لافتة عليا موحدة بها شعار البرنامج و إسمه.

نقوم بإنشاء إطار جديد ونضع فيه المكونات المطلوبة كما في الشكل (7).

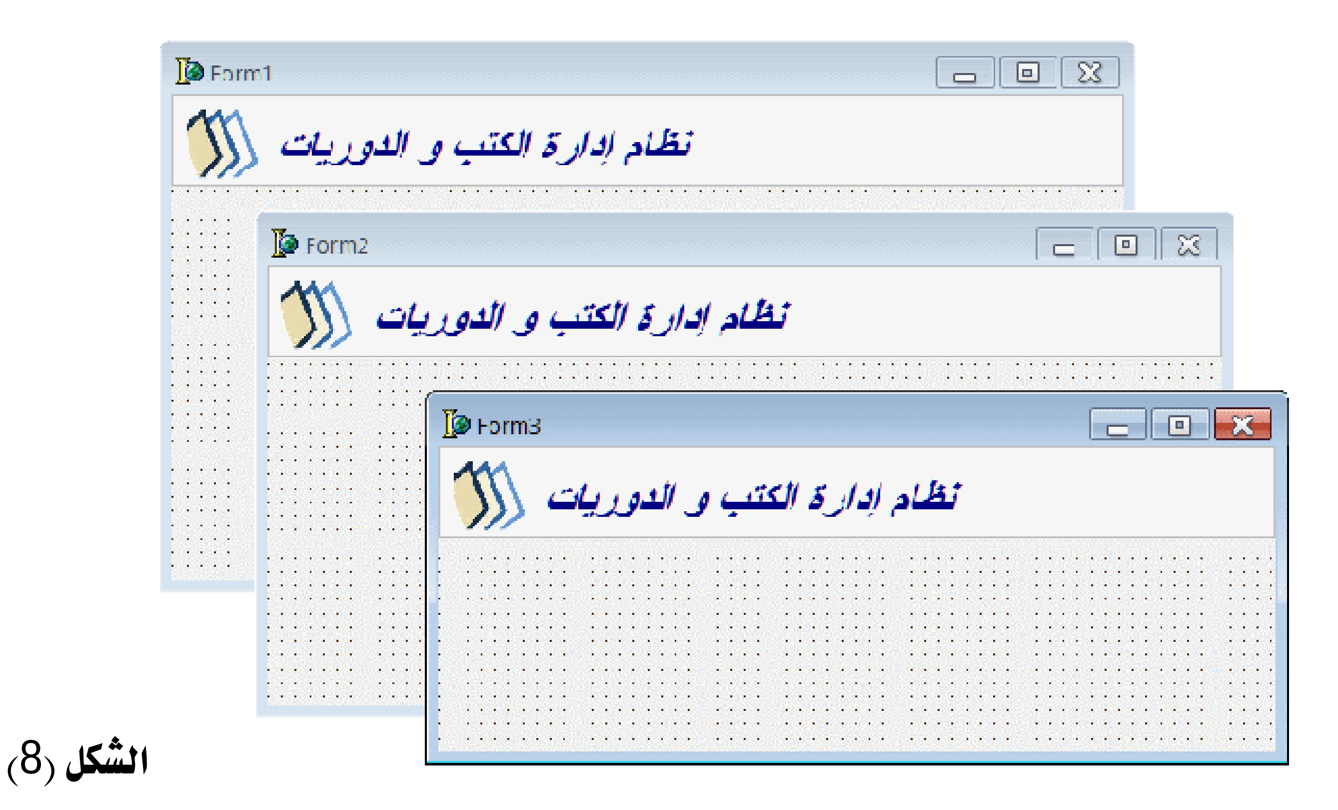


الشكل (7)

الإطار به مكون Timage و مكون Tlabel.

نسمي الإطار FramBanner ثم نقوم بحفظه بإسم fmBanner.pas . لاحظ أن ملف الإطار لايختلف كثيرا عن ملف أي Form في دلفي، في الواقع أنهما ينتميان لنفس العائلة. ولو تفحصنا ملف تعليمات الإطار سنجده يتشابه مع تلك التي لل Form.

الآن بعد حفظنا للإطار يمكننا استخدامه على نماذج الشاشات لدينا. نضع الإطار على النموذج، ونجعل من خاصية Align = alTop ليكون أعلى الشاشة. كما في الشكل (8).

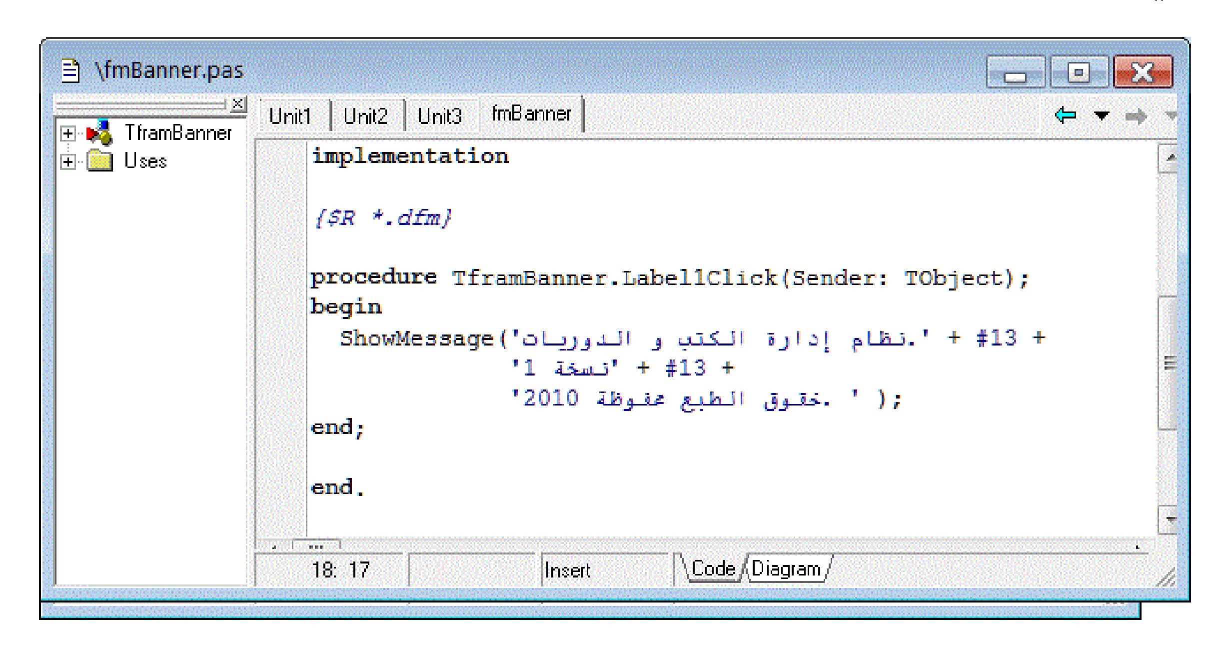


الشكل (8)

كما أشرنا أن جميع تجسّدات الإطار ترث نفس خصائص الإطار الأصلي، كما أنها تستجيب وقتيا لأي تغيير يطرأ على الإطار الأصلي ؛ ما لم نغيّر في إحدى خصائص مكوناته، فإن تلك الخاصية تحتفظ بقيمتها المعدلة، ونستطيع إرجاعها بإعادة المكون إلى صيغته الموروثة بالأمر Revert to Inherited انظر الشكل (6).

قليل من البرمجة

يمكننا إضافة تعليمات برمجية غلى الإطار, مثلا في الحدث OnClick لمكون Label1 يمكننا إظهار مربع رسالة كما في الشكل (9).



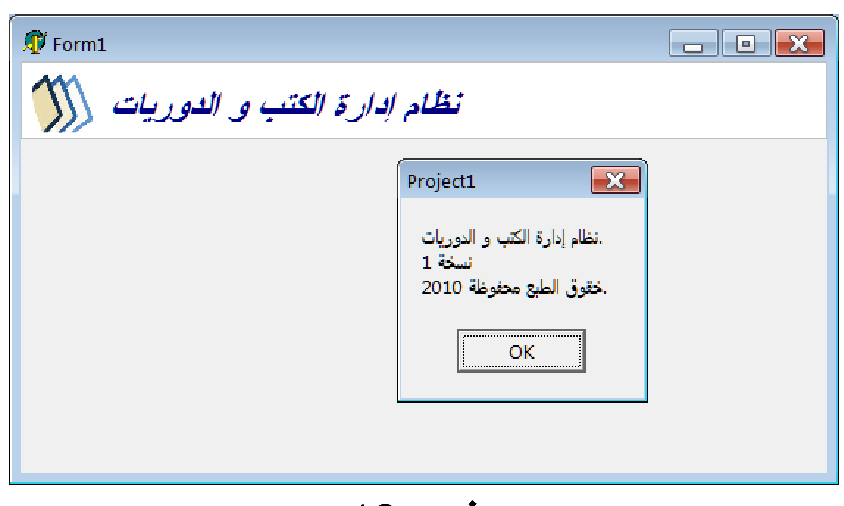
```
implementation

{$R *.dfm}

procedure TframBanner.Label1Click(Sender: TObject);
begin
  ShowMessage('نظام إدارة الكتب و الدوريات.' + #13 +
              'نسخة 1' + #13 +
              'حقوق الطبع محفوظة 2010. ' );
end;

end.
```

عندها فإن كل التجسدات لهذا الإطار يمكنها إظهار الرسالة عند النقر على مكون الكتابة. كما في الشكل (10)



الشكل (10)

التعليمات التي كتبناها محصورة فقط في الإطار الأصلي، أي أنه في نماذج الشاشات التي تستخدم هذا الأطار سوف لن تظهر هذه التعليمات. لكن يمكننا معالجة أي حدث لأي مكون في تجسدات الإطار. مثلا، لو ذهبنا إلى أي نموذج وضعنا فيه إطار، نجد أن الحدث OnClick لمكون الكتابة غير مخصص، و لكننا إذا استدعيناه ستقوم دلفي تلقائيا بالتأكيد على مناداة التعليمات في الإطار الأصلي. مثل التالي:

```
procedure TForm1.framBanner1Label1Click(Sender: TObject);
begin
  framBanner1.Label1Click(Sender);
end;
```

الآن إذا قمنا بإلغاء هذه التعليمة؛ فإن التعليمات في الإطار الأصلي لهذا الحدث سوف لن تنفذ. ونستطيع أن نكتب تعليمات بديلة تنفذ فقط على مستوى هذه النسخة من الإطار، أو يمكن كتابة تعليمات قبلها لتنفذ أولا، ثم تعليمة الإطار الأصلية ثم تنفيذ تعليمات أخرى لاحقة كالتالي:

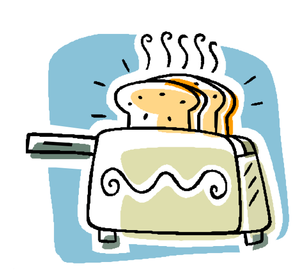
```
procedure TForm1.framBanner1Label1Click(Sender: Tobject);
begin
  framBanner1.Color := clRed;
  framBanner1.Label1Click(Sender);
  framBanner1.Color := clWhite;
end;
```

حيث قمنا بتغيير لون الإطار للأحمر، ثم استدعاء التعليمة الأصلية لإظهار الرسالة، ثم إعادة الإطار إلى اللون الأبيض.

ملخص

تقدم الإطارات Frames صيغة قوية لإنشاء مجمعات من المكونات المترابطة التي يمكن إعادة استخدامها أكثر من مرة. الإطار يمكن أن يكون تجميعا بسيطا لعدد محدود من المكونات، أو صيغة معقدة لمجموعة من المكونات و التعليمات. يمكن أيضا استخدام مكونات قواعد البيانات داخل هذه الإطارات وربطها بالسياق العام للبرنامج. راجع (المجلد \Demos\Frames\Db) ضمن حزمة دلفي.

مكـــــونات: KOL-MCK



تعتبر هذه المكونات من ابرز المكونات التي تعالج مشكل كبر حجم الملفات التنفيذية في دلفي، حيث باستعمالها نحصل على ناتج بحجم صغير جدا، مع الحفاظ على معظم مهام المكتبات الجاهزة VCL.

تدعم كل إصدارات دلفي إلى إصدار 2010 **الحالي مع اخذ بعين الاعتبار الـــ** Unicode.

قام فريق منتدى دلفي للعرب بعمل نسخة Precompiled library **تخص إصدار دلفي** 7 **و دلفي** 2010 **و برمجة أداة تسمح للمستعمل بتثبيت أو إلغاء تثبيت مكونات** KOL-MCK **بدقة و سهولة متناهية.**

طريقة استعمال المكونات:

يجب أن ننتبه إلى أنه لا يمكننا وضع مكون من القائمة مباشرة على الـــ Form **الافتراضية لدلفي دون عمل بعض التعديلات عليها، لذا من أفضل و أسهل الطرق الاعتماد على** KOL-MCK Wizard **في ذلك.**

على دلفي 7:

نذهب إلي File **ثم** New **ثم** Other **و في النافذة التي تظهر نذهب إلى تبويب** Wizards **و نختار** New KOL-MCK Project، **و بعد هذه العملية يطلب منا القيام بحفظ ملفات المشروع، و هنا تنتهي عملية الإنشاء و يمكننا استغلال كل مكونات** KOL-MCK **دون مشاكل.**

على دلفي 2010:

نذهب إلي File **ثم** New **ثم** New KOL-MCK Project **و نقوم بنفس عملية الحفظ للمشروع المذكورة فوق.**

ملاحظة: **التعامل مع مكونات** KOL-MCK **يختلف عن التعامل مع** VCL **العادية من حيث خصائص المكونات المعروفة، مثلا مكون** Memo **لا تحتوي على خاصية** Lines.

مثال: **من** TStringList **إلى** PStrList

```
var StrList: TStringList;
begin
  StrList := TStringList.Create;
  StrList.Add('Delph4Arab');
  StrList.Add('www.delphi4arab.com');
  StrList.SaveToFile(ExtractFilePath(ParamStr(0)) + 'Test.txt');
  StrList.Free;
end;
```

TStringList

```
var StrList: PStrList;
begin
  StrList := NewStrList;
  StrList.Add('Delph4Arab');
  StrList.Add('www.delphi4arab.com');
  StrList.SaveToFile(ExtractFilePath(ParamStr(0)) + 'Test.txt');
  StrList.Free;
end;
```

PStrList

مكونـــــــــــات دلفي

مكـــــون: ActionList

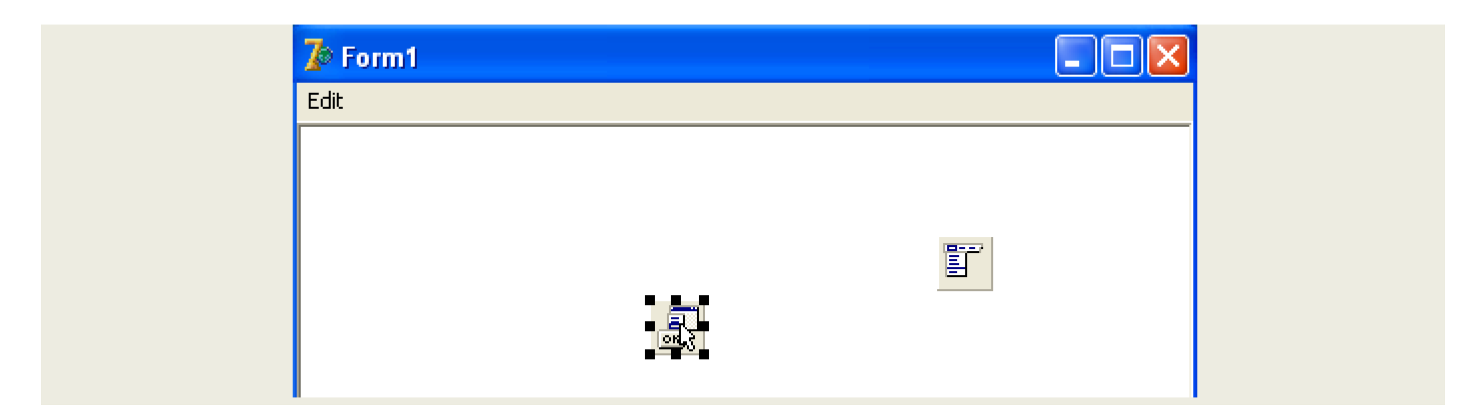


من مزايا دلفي هي المكونات الجاهزة الاستعمال و التي لا تحتاج إلى كتابة سطر واحد من الأوامر، و من هذه المكونات، مكون ActionList الذي يحتوي على مجموعة كبيرة من الـ Actions المستعملة في اغلب الأحيان.

نجد مكون ActionList في تبويب Standard.

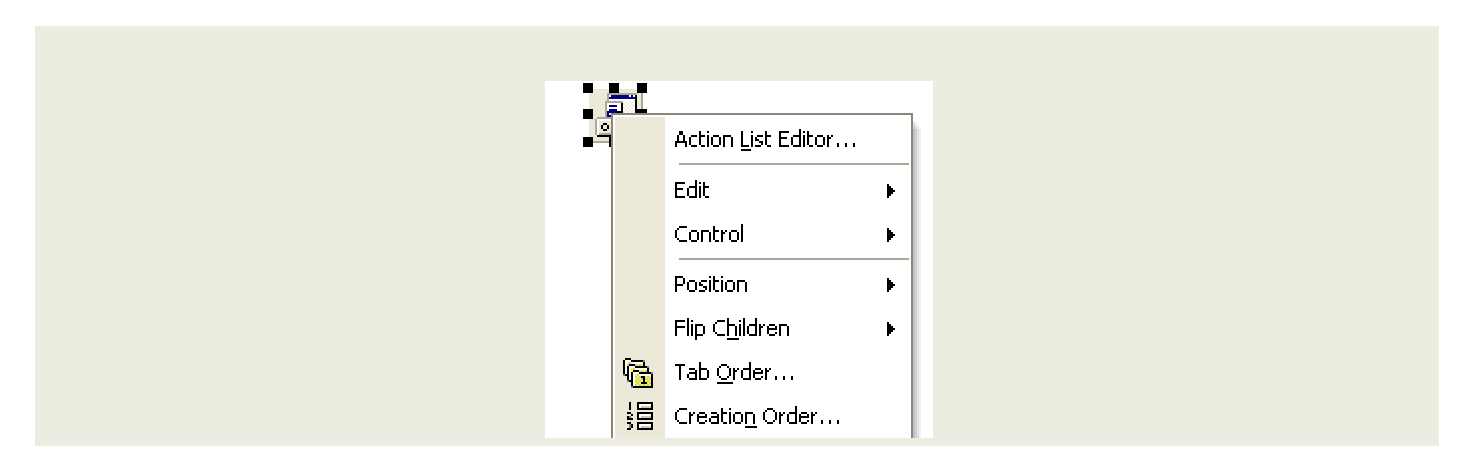


ننشئ مشروع تجريبي و نضيف إليه مكونات MainMenu، Memo و ActionList.

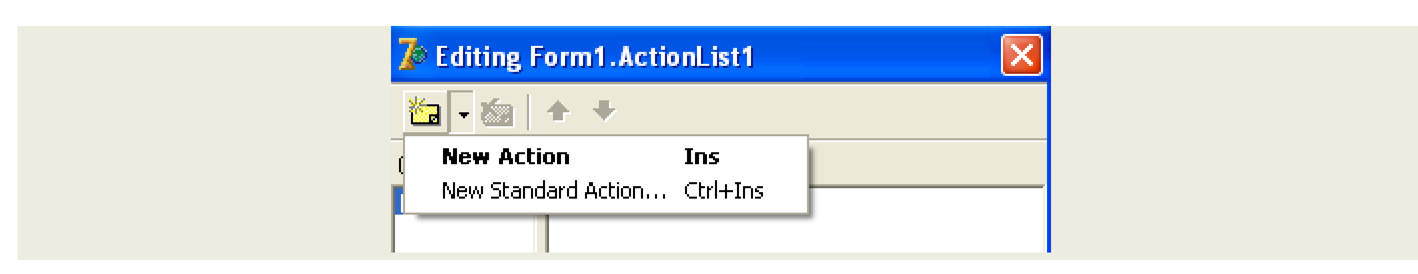


بالنسبة لمكون MainMenu نقوم بإضافة قائمة باسم Edit تحتوي على عنصرين Copy و Paste

نختار المكون ActionList و نقوم بالنقر على الزر الأيمن للفأرة ثم نختار ActionList Editor.

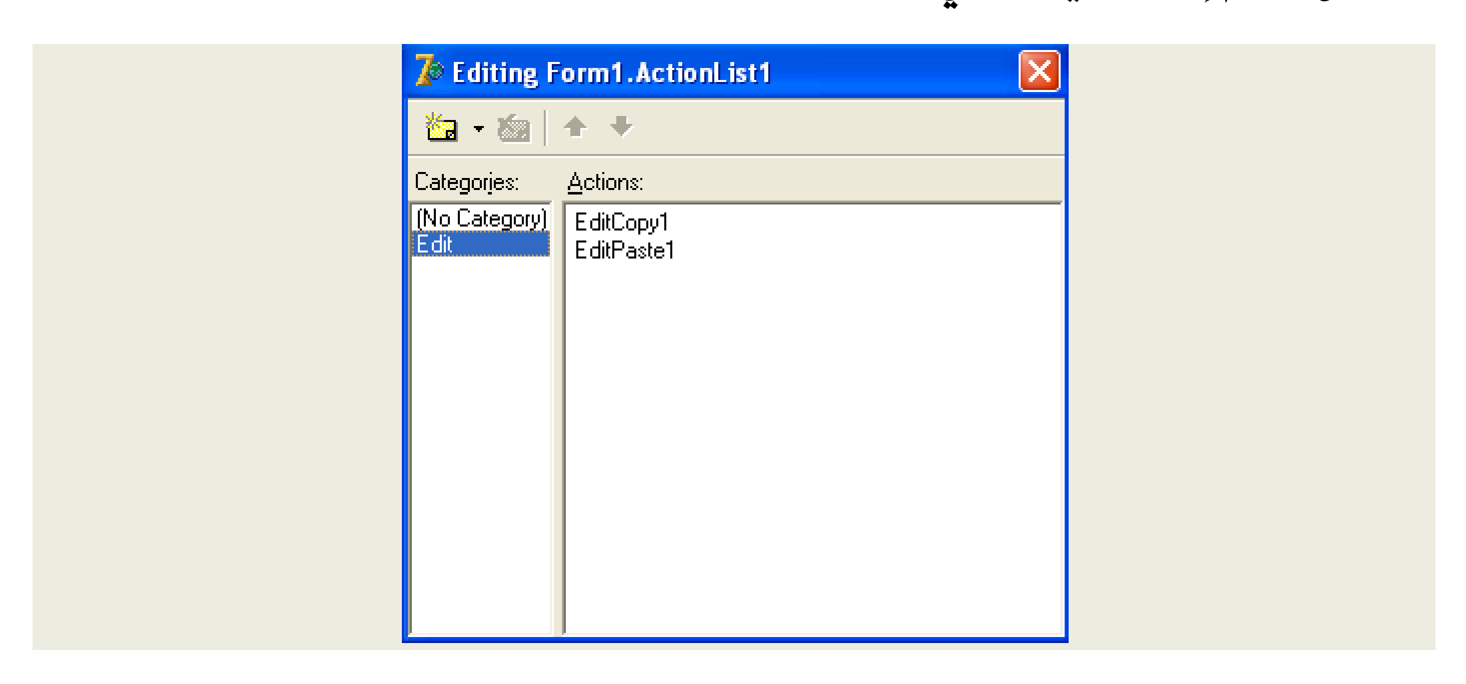


نذهب إلى New Standard Action.

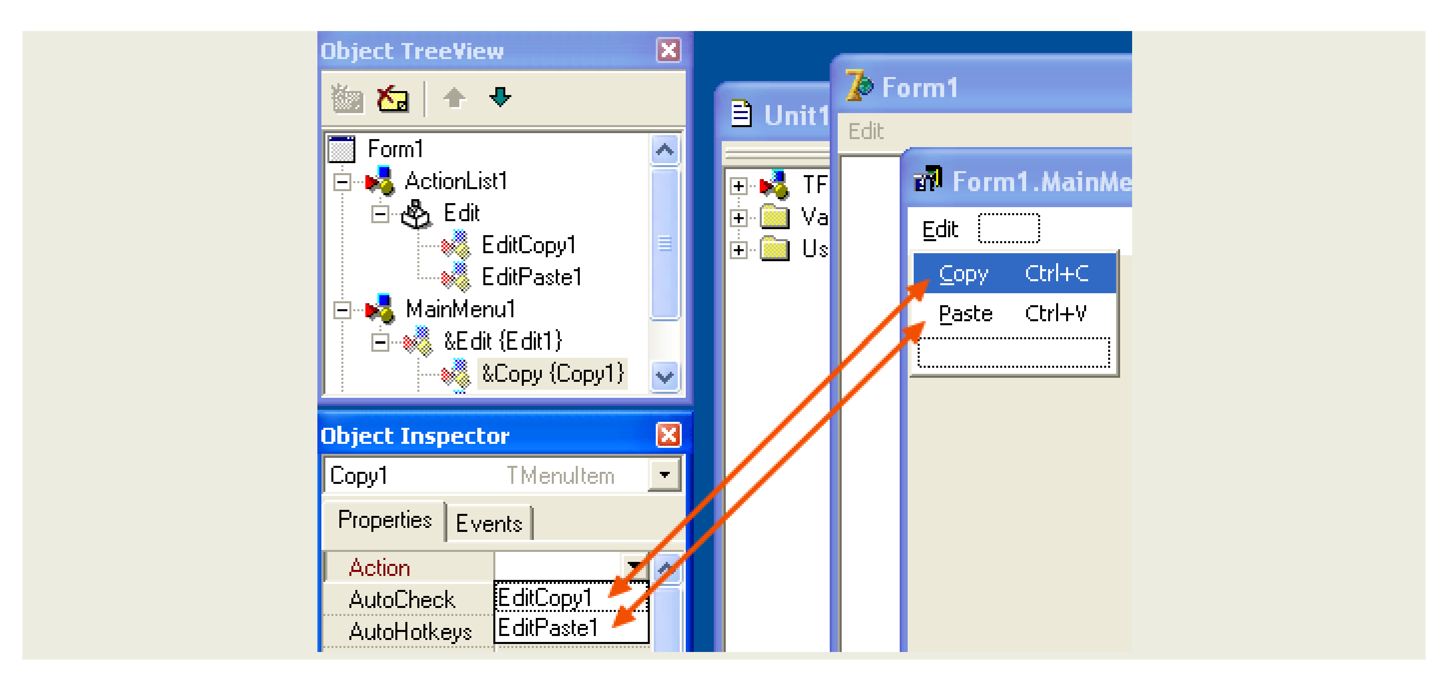


نختار إضافة TEditCopy/TEditPaste action من القائمة.

تتأكد من انه تم إضافة الخيارات التي اخترناها



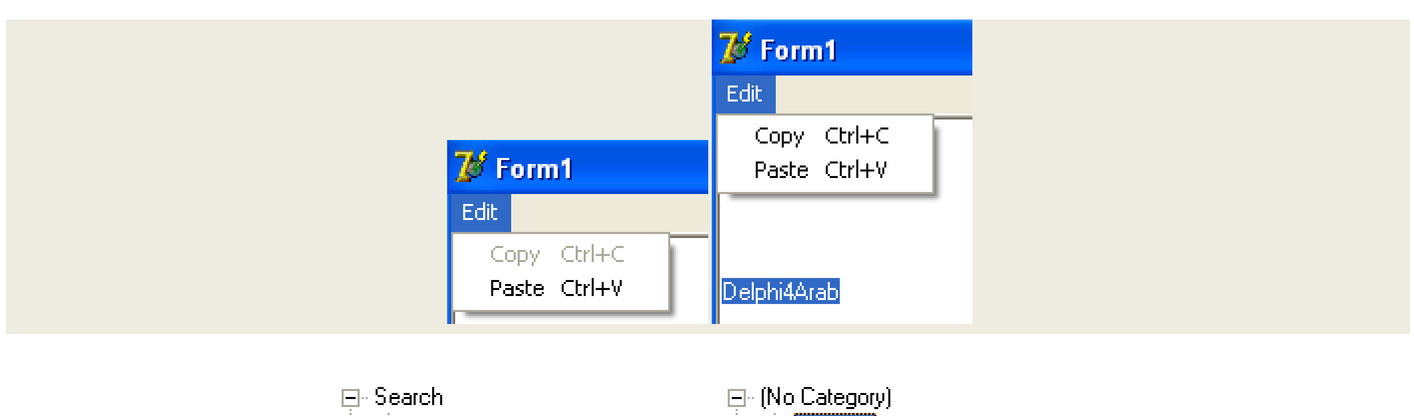
الآن نذهب إلى مكون MainMenu و نتوجه إلى قائمة جديدة Edit تحتوي على عناصر Copy/Paste



و نذهب إلى خصائص كل عنصر و في خيار Action نرفع الـــ Action المناسبة للمثال التجريبي.

```
Copy   à Action := EditCopy1
Paste  à Action := EditPast1
```

بعد تجربة تشغيل المشروع نلاحظ أن عملية النسخ و اللصق تشتغل بنجاح دون كتابة أي أمر الـ Events.



- Search
  - TSearchFind
  - TSearchFindNext
  - TSearchReplace
  - TSearchFindFirst
- Tab
  - TPreviousTab
  - TNextTab
- List
  - TListControlCopySelection
  - TListControlDeleteSelection
  - TListControlSelectAll
  - TListControlClearSelection
  - TListControlMoveSelection
  - TStaticListAction
  - TVirtualListAction
- Dialog
  - TOpenPicture
  - TSavePicture
  - TColorSelect
  - TFontEdit
  - TPrintDlg
- Internet
  - TBrowseURL
  - TDownLoadURL
  - TSendMail
- Dataset
  - TDataSetFirst
  - TDataSetPrior
  - TDataSetNext
  - TDataSetLast
  - TDataSetInsert
  - TDataSetDelete
  - TDataSetEdit
  - TDataSetPost
  - TDataSetCancel
  - TDataSetRefresh
  - TClientDataSetApply
  - TClientDataSetRevert
  - TClientDataSetUndo
- Tools
  - TCustomizeActionBars
- (No Category)
  - TAction
- Edit
  - TEditCut
  - TEditCopy
  - TEditPaste
  - TEditSelectAll
  - TEditUndo
  - TEditDelete
- Format
  - TRichEditBold
  - TRichEditItalic
  - TRichEditUnderline
  - TRichEditStrikeOut
  - TRichEditBullets
  - TRichEditAlignLeft
  - TRichEditAlignRight
  - TRichEditAlignCenter
- Help
  - THelpContents
  - THelpTopicSearch
  - THelpOnHelp
  - THelpContextAction
- Window
  - TWindowClose
  - TWindowCascade
  - TWindowTileHorizontal
  - TWindowTileVertical
  - TWindowMinimizeAll
  - TWindowArrange
- File
  - TFileOpen
  - TFileOpenWith
  - TFileSaveAs
  - TFilePrintSetup
  - TFilePageSetup
  - TFileRun
  - TFileExit

قائمة بكل الـ Actions الموجودة في مكون ActionList

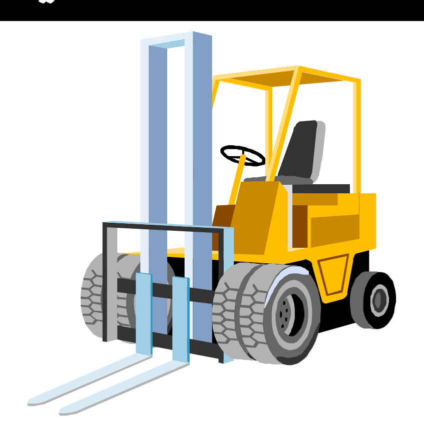
أوامـــــــر دلفــــــي – بقلم STRELiTZIA



iif (immediate if)

ضـــــــد

if...then...else



استزادا من لغات برمجة أخرى مثل Visual Basic سنحاول استبدال الأمر الشرطي المعروف if.then.else بأمر شرطي آخر و هو معرف أيضا باسم immediate if و يكتب iif بإضافة i إلى if.

هيكل الدالة:

```
function iif(
   Expression: boolean,
   TruePart: type,
   FalsePart: type
): type;
```

Expression : التعبير الشرطي أو Condition و هو من نوع Boolean
TruePart: القيمة المرجعة في حالة تحقق الشرط و نوعها غير معين.
FalsePart: القيمة المرجعة في حالة عدم تحقق الشرط و نوعها غير معين.

طريقة الكتابة الشرطية العادية if..then..else:

```
if a = b then
 Label1.Caption := 'توجد مساواة'
 else
 Label1.Caption := 'لا توجد مساواة';
```

طريقة الكتابة الشرطية باستعمال دالة immediate if:

```
Label1.Caption := iif(a = b, 'توجد مساواة', 'لا توجد مساواة');
```

تعريف الدالة و إعطائها نوع string:

```
function iif(Expression: boolean; TruePart, FalsePart: string): string;
begin
  if Expression then
    Result := TruePart
  else
    Result := FalsePart;
end;
```

تعريف الدالة و إعطائها نوع integer:

```
function iif(Expression: boolean; TruePart, FalsePart: integer): integer;
begin
  if Expression then
    Result := TruePart
  else
    Result := FalsePart;
end;
```

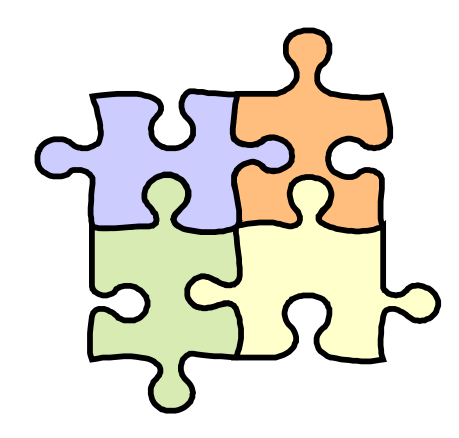
طريقة الاستدعاء:

```
  ShowMessage(iif(TestEdt.Text = '123', 'Sucess', 'Error'));

  ShowMessage(IntToStr(iif(TestEdt.Text = '123', 1, 0)));
```

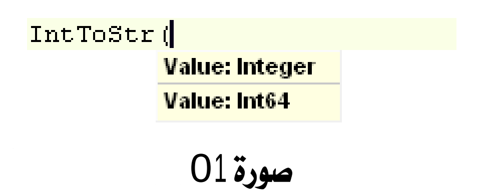
يمكننا كتابة مجموعة من الدوال تدعم أنواع مختلفة من المتغيرات و تحمل نفس اسم الاستدعاء و جعل الـ Compiler الخاص بدلفي هو الذي يختار دالة المعالجة حسب نوع البراميترات الممررة، يكفي فقط إضافة overload directive إلى دوال المعالجة. (المقال التالي يتحدث عن الفكرة)

Directive: Overload



في بعض الحالات نحتاج معالجة معطيات من أنواع مختلفة مثل نوع string، integer أو غيرها عن طريق دالة واحدة دون الحاجة إلى تغيير اسم استدعاء الدالة إلى iifString, iifInteger, iifReal مثلا...، و الحل يكمن في directive تجعل الـــ Compiler الخاص بدلفي يختار حسب نوع البراميتر المررالدالة المناسبة لمعالجة المعطيات حتى و إن كانت تحمل نفس اسم الاستدعاء، هذه الـ directive تسمى overload.

مثال توضيحي: من وحدة SysUtils نأخذ دالة التحويل IntToStr التي نستعملها في اغلب الأحيان و هي تتقبل نوعين مختلفين من البراميترات Integer و Int64، و يظهر خيار البراميتر تلقائيا عند فتح قوس إدخال البراميترات. صورة 01 : نلاحظ ظهور نوعين مختلفين.



صورة 01

و لو اطلعنا على محتوى وحدة SysUtils سوف نلاحظ أن كثير من الدوال تحمل نفس اسم الاستدعاء، و الخيار بينها يقوم به الـــ Compiler بفحص نوع البرامير أولا قبل استدعاء الدالة، إذا كان النوع integer يستدعي الدالة الأولى و إن كان النوع Int64 يستدعي الدالة الثانية. صورة 02

```
function IntToStr(Value: Integer): string; overload;
function IntToStr(Value: Int64): string; overload;
```

صورة 02

و لفهم العملية بصورة تطبيقية ننشئ مشروع جديد ثم نأخذ نفس المثال الخاص بدالة iif. صورة 03 و 04

```
function iif(Expression: boolean; TruePart, FalsePart: string): string; overload;
begin
  if Expression then
    Result := TruePart
  else
    Result := FalsePart;
end;
```

صورة 03

```
function iif(Expression: boolean; TruePart, FalsePart: integer): integer; overload;
begin
  if Expression then
    Result := TruePart
  else
    Result := FalsePart;
end;
```

صورة 04

في حدث OnClick الخاص بالزر TestBtn نضيف طريقة الاستدعاء. صورة 05

```
  ShowMessage(iif(TestEdt.Text = '123', 'Sucess', 'Error'));

  ShowMessage(IntToStr(iif(TestEdt.Text = '123', 1, 0)));
```

صورة 05

نضع نقطة توقف على إجراء ShowMessage الأول و الثاني ثم نشغل المشروع و ننقر على الزر TestBtn و عند توقف دلفي نقوم بالتتبع باستعمال مفتاح F7. صورة 06

```
procedure TForm1.TestBtnClick(Sender: TObject);
begin
  ShowMessage(iif(TestEdt.Text = '123', 'Sucess', 'Error'));
  ShowMessage(IntToStr(iif(TestEdt.Text = '123', 1, 0)));
end;
```

صورة 05

نلاحظ أن الـ Compiler بعد فحصة للبراميترات يستدعي في كل مرة الدالة التي تقبل نوع البراميتر الممرر.

برامج لها علاقة بدلفي – بقلم STRELiTZIA

Interactive Delphi Reconstructor



نبذة عن الأداة:

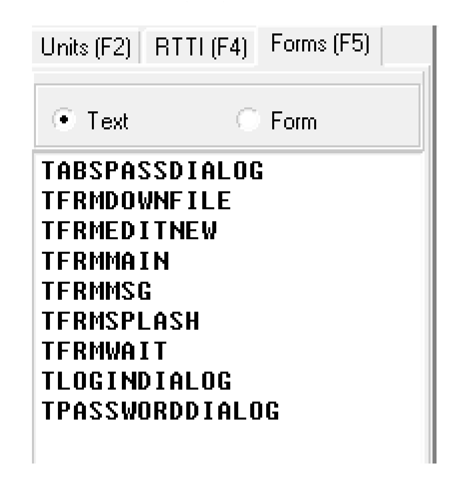
IDR **هي عبارة عن أداة قيد التطوير من برمجة** crypto **لتفكيك الملفات التنفيذية المبرمجة بدلفي لغرض تحليلها و الحصول على اكبر عدد من المعلومات عن الملف المفكوك.**

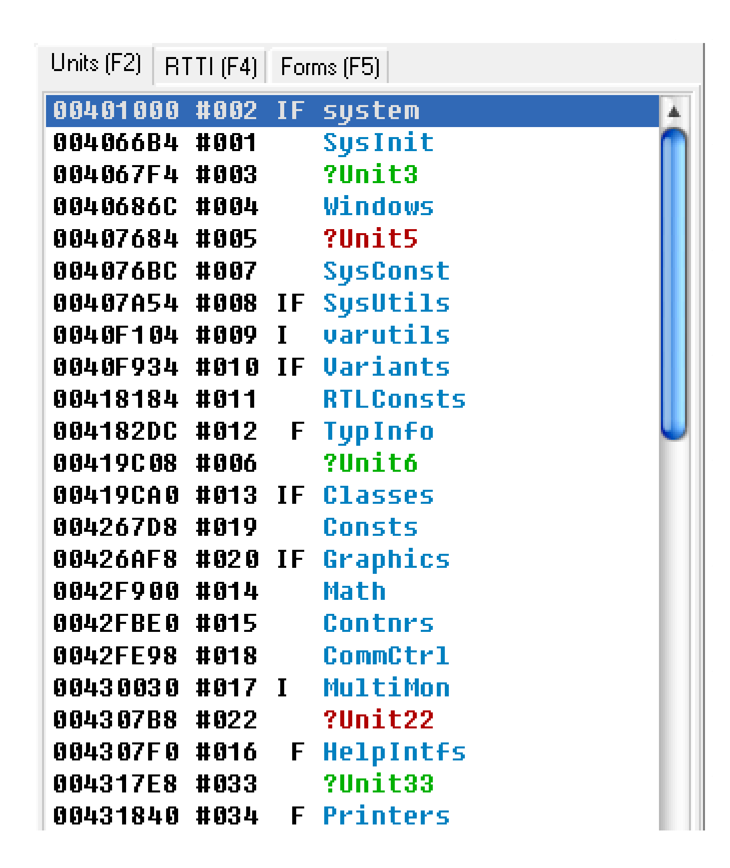
تعتبر الخلف الجيد للأداة المشهورة DeDe **بعد توقف مبرمجها عن تطويرها.**

رابط التحميل:

http://kpnc.org/idr32/en/index.htm

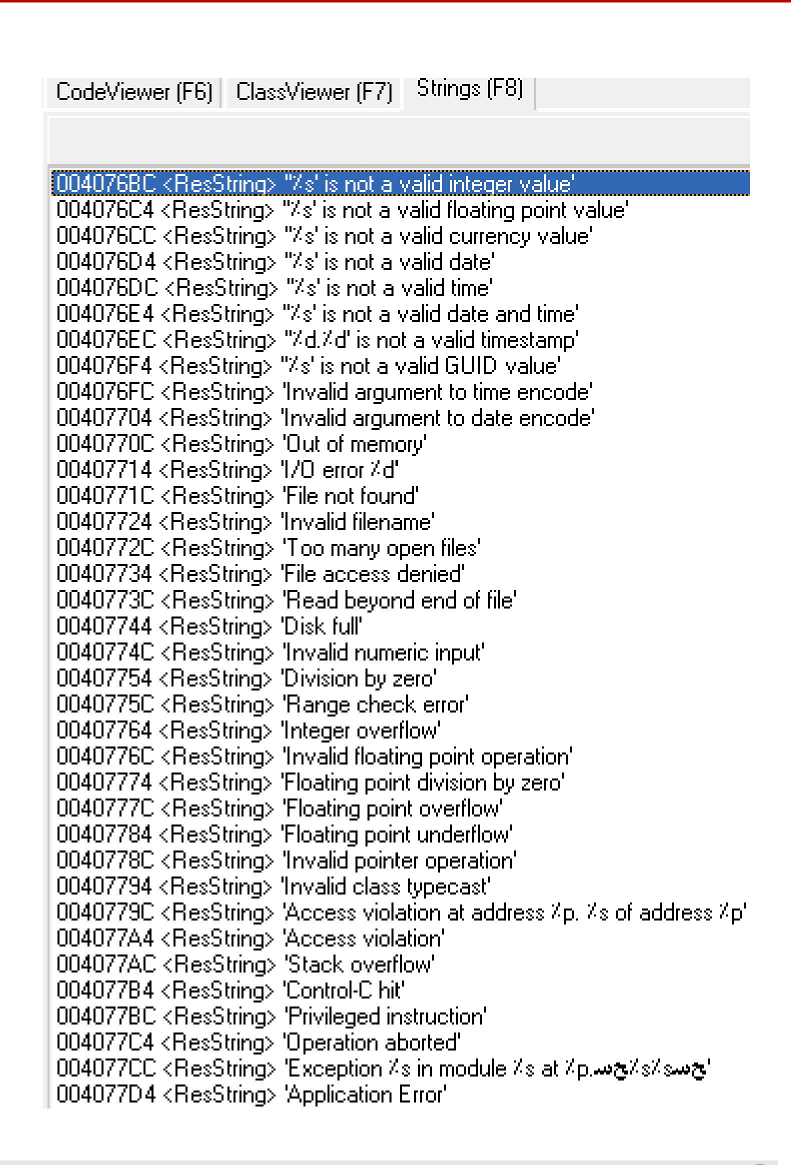
```
EntryPoint
0053C904    push    ebp
0053C905    mov     ebp,esp
0053C907    add     esp,0FFFFFFF0
0053C90A    push    ebx
0053C90B    push    esi
0053C90C    mov     eax,53C584; gvar_0053C584:Pointer
0053C911    call    @InitExe
0053C916    mov     ebx,dword ptr ds:[54AB9C]; ^Application:TApplication
0053C91C    mov     esi,dword ptr ds:[54AC3C]; ^gvar_0054DA90:TFrmSplash
0053C922    mov     ecx,dword ptr [ebx]
0053C924    mov     dl,1
0053C926    mov     eax,[005380E8]; TFrmSplash
0053C92B    call    TCustomForm.Create; TFrmSplash.Create
0053C930    mov     dword ptr [esi],eax
0053C932    mov     eax,dword ptr [esi]
0053C934    call    TCustomForm.Show
0053C939    mov     eax,dword ptr [esi]
0053C93B    mov     edx,dword ptr [eax]
0053C93D    call    dword ptr [edx+88]; TWinControl.Update
>0053C943   jmp     0053C94C
0053C945    mov     eax,dword ptr [ebx]
0053C947    call    TApplication.ProcessMessages
0053C94C    mov     eax,dword ptr [esi]
0053C94E    mov     eax,dword ptr [eax+2FC]
0053C954    cmp     byte ptr [eax+40],0
<0053C958   jne     0053C945
0053C95A    mov     eax,dword ptr [ebx]
0053C95C    call    TApplication.Initialize
0053C961    mov     eax,dword ptr [ebx]
```





Units (F2) RTTI (F4) Forms (F5)

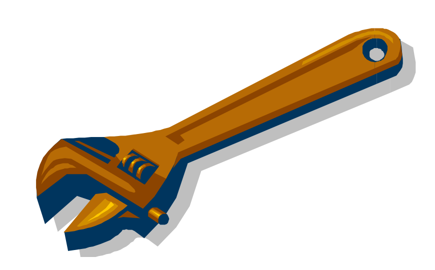
```
00401000 <Enumeration> Boolean
00401028 <Char> Char
0040103C <Integer> Smallint
00401054 <Integer> Integer
0040106C <Integer> Byte
00401080 <Integer> Word
00401094 <Integer> Cardinal
004010AC <Int64> Int64
004010C8 <Float> Double
004010D8 <Float> Currency
004010E8 <AnsiString> String
004010F4 <WideString> WideString
00401104 <Variant> Variant
00401168 <Class> TObject
00401188 <Interface> IInterface
00401268 <DynArray> TBoundArray
00401290 <Float> TDateTime
0040686C <Integer> HWND
00406880 <Enumeration> ?Unit4.1
00406930 <Set> TOwnerDrawState
00407A54 <AnsiString> TFileName
00407A64 <Record> TSearchRec
00407AEC <Class> Exception
004084F4 <DynArray> ?Unit8.74
```


CodeViewer (F6)
ClassViewer (F7)
Strings (F8)
004076BC <ResString> ''%s' is not a valid integer value'
004076C4 <ResString> ''%s' is not a valid floating point value'
004076CC <ResString> ''%s' is not a valid currency value'
004076D4 <ResString> ''%s' is not a valid date'
004076DC <ResString> ''%s' is not a valid time'
004076E4 <ResString> ''%s' is not a valid date and time'
004076EC <ResString> ''%d.%d' is not a valid timestamp'
004076F4 <ResString> ''%s' is not a valid GUID value'
004076FC <ResString> 'Invalid argument to time encode'
00407704 <ResString> 'Invalid argument to date encode'
0040770C <ResString> 'Out of memory'
00407714 <ResString> 'I/O error %d'
0040771C <ResString> 'File not found'
00407724 <ResString> 'Invalid filename'
0040772C <ResString> 'Too many open files'
00407734 <ResString> 'File access denied'
0040773C <ResString> 'Read beyond end of file'
00407744 <ResString> 'Disk full'
0040774C <ResString> 'Invalid numeric input'
00407754 <ResString> 'Division by zero'
0040775C <ResString> 'Range check error'
00407764 <ResString> 'Integer overflow'
0040776C <ResString> 'Invalid floating point operation'
00407774 <ResString> 'Floating point division by zero'
0040777C <ResString> 'Floating point overflow'
00407784 <ResString> 'Floating point underflow'
0040778C <ResString> 'Invalid pointer operation'
00407794 <ResString> 'Invalid class typecast'
0040779C <ResString> 'Access violation at address %p. %s of address %p'
004077A4 <ResString> 'Access violation'
004077AC <ResString> 'Stack overflow'
004077B4 <ResString> 'Control-C hit'
004077BC <ResString> 'Privileged instruction'
004077C4 <ResString> 'Operation aborted'
004077CC <ResString> 'Exception %s in module %s at %p.'
004077D4 <ResString> 'Application Error'


About...
Application: IDR by crypto
Version: 2.1.1.33
My respects to:
DaFixer for wonderful program DeDe
Alexey Hmelnov for nice utility dcu32int and open source code
SVLab for continual IDR testing, help and site creation
sendersu for IDR FormViewer assistance
Kris Kaspersky for IDR hosting
E-mail:
Homepage:
OK

حل تمرين العدد 01 – بقلم STRELiTZIA

نظرة على برمجة Delphi Wizard/Expert



كان المطلوب في تمرين العدد رقم 01 من مجلة دلفي للعرب، محاولة برمجة أداة تدمج في واجهة التطوير الخاصة بدلفي.

حل التمرين:

بالنسبة لشخص الذي يقوده حب الاستطلاع إلى تصفح مجلدات التنصيب الخاصة بدلفي سيكتشف ما يسره من أمثلة و وحدات جاهزة الاستعمال تعالج معظم ميادين البرمجة، و من ضمنها وحدة أو وحدات تحتوي على دوال تسمح لنا بدمج أدوات في واجهة التطوير.

وحدة ExptIntf.pas:

مسارها الافتراضي:

```
X:\Program Files\Borland\Delphi7\Source\ToolsAPI
X:\Program Files\Embarcadero\RAD Studio\7.0\source\ToolsAPI
```

تحتوي على مجموعة من تعريفات الدوال التي نحتاجها في هذا التطبيق.

```
public
    { Expert UI strings }
    function GetName: string; virtual; stdcall; abstract;
    function GetAuthor: string; virtual; stdcall; abstract;
    function GetComment: string; virtual; stdcall; abstract;
    function GetPage: string; virtual; stdcall; abstract;
{$IFDEF MSWINDOWS}
    function GetGlyph: HICON; virtual; stdcall; abstract;
{$ENDIF}
{$IFDEF LINUX}
    function GetGlyph: Cardinal; virtual; stdcall; abstract;
{$ENDIF}
    function GetStyle: TExpertStyle; virtual; stdcall; abstract;
    function GetState: TExpertState; virtual; stdcall; abstract;
    function GetIDString: string; virtual; stdcall; abstract;
    function GetMenuText: string; virtual; stdcall; abstract;

    { Launch the Expert }
    procedure Execute; virtual; stdcall; abstract;
```

التعريف بالدوال:

```
{
    This is the declaration of the pure-virtual base class for the expert
    interface within the Delphi IDE.

    NOTE: In Delphi 1.0, the GetGlyph function used to return an HBITMAP,
          whereas now it must return an HICON.

    GetName -        REQUIRED.  This must return a unique descriptive name
                     identifying this expert.

    GetAuthor  -     REQUIRED is style is esForm or esProject.  This should
                     return the "author" of this add-in/expert.  This could
                     be a person or company, for example.  This value will
                     be displayed in the Object Repository.

    GetComment -     REQUIRED if style is esForm or esProject.  This should
                     return a 1 - 2 sentence describing the function of this
                     expert.

    GetPage -        REQUIRED if style is esForm or esProject. Should return
                     short string indicating on  which page in the repository
                     this expert should be placed.  NOTE: The user can still
                     override this setting from the Tool|Repository dialog.

    GetGlyph -       REQUIRED if style is esForm or esProject.  This should
                     return a handle to a icon to be displayed in the form or
                     project list boxes or dialogs.  Return 0 to display the
                     default icon.

    GetStyle -       REQUIRED.  Returns one of four possible values:
                     esStandard - Tells the IDE to treat the interface to
                                  this expert as a menu item on the Help
                                  menu.
                     esForm     - Tells the IDE to treat this expert interface
                                  in a fashion similar to form templates.
                     esProject  - Tells the IDE to treat this interface in a
                                  fashion similar to project templates.
                     esAddIn    - Tells the IDE that this expert handles all its
                                  own interfacing to the IDE through the
                                  TIToolServices interface.

    GetState -       REQUIRED. If the style is esStandard, esChecked will cause
                     the menu to display a checkmark. NOTE: This function is
                     called each time the expert is shown in a menu or listbox in
                     order to determine how it should be displayed.

    GetIDString -    REQUIRED. This ID string should be unique to all experts
                     that could be installed.  By convention, the format of the
                     string is:
                     CompanyName.ExpertFunction, ex. Borland.WidgetExpert

    GetMenuText -    REQURED if style is esStandard.  This should return the
                     actual text to display for the menu item. NOTE: This
                     function is called each time the parent menu is pulled-down,
                     so it is possible to provide context sensative text.

    Execute -        REQUIRED if style is esStandard, esForm, or esProject.
                     Called whenever this expert is invoked via the menu, form
```

```
                   repository dialog, or project repository dialog.  The style
                   will determine how the expert was invoked.  This procedure
                   is never called if the style is esAddIn.

    TExpertInitProc - defines the number and types of parameters passed to the
                      single exported entry-point to the expert DLL.
                      ToolServices - a pure-virtual class containing all the
                                     tool services provided by the IDE.
                      RegisterProc - The function to call in order to register
                                     an expert.  NOTE: This function is called
                                     once for each expert instance that the DLL
                                     wants to register with the IDE.
                      Terminate -    Set this parameter to point to a procedure
                                     that will be called immediately before the
                                     expert DLL is unloaded by the IDE.  Leave
                                     nil, if not needed.
  }
```

وحدة المشروع WizardTestProject:

1ـ نقوم بإنشاء وحدة جديدة باستعمال أداة Notpad مثلا و نعرفها بهذه الطريقة:

```
unit WizardTestProject;
interface
uses
  Windows, ExptIntf;

Type
  TWizardTest = class(TIExpert)
  public

    function GetStyle: TExpertStyle; override;
    function GetIDString: string; override;
    function GetName: string; override;
    function GetMenuText: string; override;
    function GetState: TExpertState; override;
    procedure Execute; override;
  end;

  procedure Register;

implementation
```

جزء قبل implementation

```
uses MainFrm ;
```

2ـ ثم نضيف تعريف لوحدة MainFrm و هي عبارة عن Unit مع Form مستقلة فارغة لا تحتوي على أوامر وضعت كمثال و التي سوف سيتم استدعاءها في حدث Execute لاحقا.

3- و أخيرا نكمل بإضافة الإجراءات و الدوال التالية:

```
procedure Register;
begin
    RegisterLibraryExpert(TWizardTest.Create)
end;

function TWizardTest.GetStyle: TExpertStyle;
begin
    Result := esStandard;
end;

function TWizardTest.GetIDString: String;
begin
    Result := 'First.TWizardTest';
end;

function TWizardTest.GetName: String;
begin
    Result := 'Wizard Test Sample';
end;

function TWizardTest.GetMenuText: String;
begin
    Result := '&Wizard Test Sample';
end;

function TWizardTest.GetState: TExpertState;
begin
    Result := [esEnabled];
end;

procedure TWizardTest.Execute;
begin
    with TWizardExpertTest.Create(nil) do
    begin
      ShowModal;
      Free;
   end;
end;
end.
```

4ـ بعد الانتهاء من حفظ الوحدة باسم WizardTestProject بامتداد pas طبعا نقوم بإنشاء MainFrm – صورة 01

MainFrm

```
interface

uses
  Windows,
  Messages,
  SysUtils,
  Classes,
  Graphics,
  Controls,
  Forms,
  Dialogs,
  StdCtrls,
  ExtCtrls;

type
  TWizardExpertTest = class(TForm)
    DLbl: TLabel;
    WizardLbl: TLabel;
    LinePnl: TPanel;
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  WizardExpertTest: TWizardExpertTest;

implementation

{$R *.dfm}

end.
```

Delphi Wizard Sample

صورة 01

5ـ نقوم بتثبيت الوحدة WizardTestProject كمكون جديد بنفس طريقة تثبيت مكونات دلفي (راجع طريقة تثبيت مكونات دلفي في مواضيع منتدى دلفي للعرب)

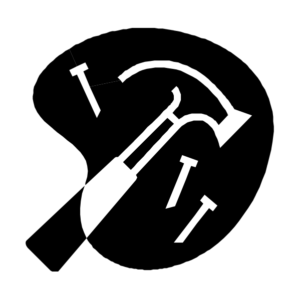
6- النتيجة:

ملاحظة: عند إنشاء وحدة Expert بخيارات افتراضية دلفي يقوم بإضافة الوحدة في جهة Help، و في حالة رغبة إضافة الوحدة في مكان محدد، Tools مثلا يجب علينا إضافة أوامر تقوم بذلك لكي لا يتم توجيه الوحدة افتراضيا بعد تثبيتها إلى جهة Help.

```
ReferenceMenuItem := MainMenu.FindMenuItem('ToolsOptionsItem');
```

تمرين هذا العدد

برمجة Flash Memory Filter/Locker



المطلوب:

برمجة تطبيق يقوم بتتبع إيصال Flash Memory بالجهاز و يظهر رسالة للمستخدم تعلمه بذلك مع خيار فتح الـ Flash Memory أو عدمه.

إضافة خيار Lock، قفل الـ Flash Memory و منع الدخول إليه.

بالتوفيق إن شاء الله